■ ชูเกียรติ วรสุชีพ

Java : จากเริ่มต้นจนถึงวันนี้ ปี 2000

พัฒนาการตั้งแต่เริ่มต้น จนถึงปัจจุบัน พร้อมเทคนิควิธีการ นำมาใช้งาน

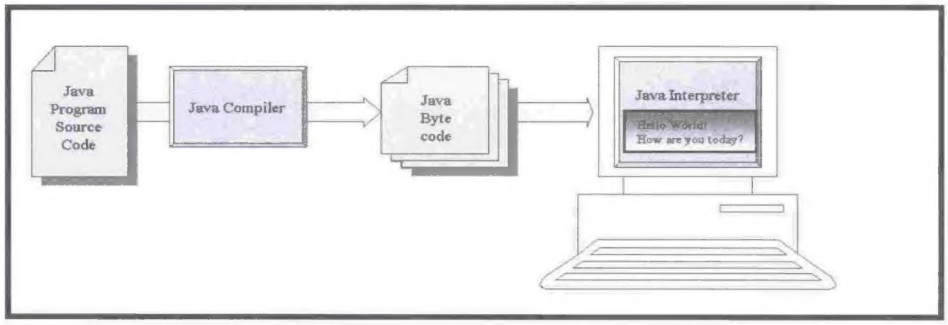
Java ชื่อนี้โด่งดังมากในวงการคอมพิวเตอร์ยุค ค.ศ. 2000 เชื่อได้ว่าทุกคนในวงการคงเคยได้ยินหรือได้ใช้มาบ้างแล้วไม่มากก็น้อย บางคนก็บอกว่าจาวานี้เยี่ยม เป็นคู่แข่งที่น่ากลัวของ Microsoft แต่บ้างก็ว่าไม่มีทางสู้กับ Microsoft อยู่ดี บ้างก็ว่าเป็นภาษาคอมพิวเตอร์ภาษาหนึ่งที่พัฒนามาจาก C หรือ C++ แต่บ้างก็ว่าไม่ใช่ และเป็นอะไรที่กว้างกว่าเพียงแค่ภาษาคอมพิวเตอร์ แต่ความจริงแล้วจาวาคืออะไร?

ก่อนที่จะไปดูเรื่องราวทางเทคนิคเกี่ยวกับจาวา ลองไปดูตัวเลขสถิติกันสักเล็กน้อยที่เกี่ยวกับจาวา ตัวเลขเหล่านี้มาจาก DICE IT Tracker ของ Earthweb ซึ่งสรุปตัวเลขเกี่ยวกับงานทางด้านจาวาในสหรัฐอเมริกาสำหรับเดือนพฤษภาคม 2000 ตัวเลขนี้มาจากผู้ที่ทำงานเกี่ยวข้องกับจาวาไม่มากก็น้อยประมาณ 4,300 ตำแหน่งจากทั้งหมดกว่า 1.3 แสนตำแหน่งงาน (ใครสนใจขอให้ไปหารายละเอียดได้จากเว็บของ Earthweb หรือผ่านทาง Gamelan ก็ได้)

อัตราค่าจ้างเฉลี่ยต่อชั่วโมงของตำแหน่งงานด้านจาวา สูงที่สุดอยู่ที่ย่านซิลิกอนวัลลี่ (ย่านทองคำสำหรับอุตสาหกรรมไฮเทคของโลกวันนี้ และอยู่ใกล้ ๆ ซานฟรานซิสโก) สูงถึง 93 เหรียญต่อชั่วโมง รองลงไปคือ ซานฟรานซิสโกที่ 92 เหรียญ นิวยอร์กที่ 89 เหรียญ ฟิลลาเดลเฟียที่ 79 เหรียญ บอสตันที่ 74 เหรียญต่อชั่วโมง เป็นต้น ต่อชั่วโมงนะครับที่อเมริกาเขาคิดค่าจ้างเป็นรายชั่วโมง จริงอยู่ค่าครองชีพที่นั่นสูง แต่ลองนึกภาพว่าอาหารมื้อทั่ว ๆ ไปของเขาอยู่ที่ 10-15 เหรียญ (หรือถ้าเป็นอาหารด่วนราคาถูก ๆ ก็อยู่ที่ 4-6 เหรียญต่อมื้อเอง)

รายได้ เขาชอบคิดเป็นปี (จะได้รวมโบนัสและอื่น ๆ ให้ครบ) ไม่ใช่เดือนอย่างบ้านเรา ที่นั่นรายได้ต่อปีของตำแหน่งงานด้านจาวาสูงสุดอยู่ที่ นิวยอร์ก ซิลิกอนวัลลี่ และซานฟรานซิสโก สูงถึงประมาณ 9 หมื่นเหรียญต่อปี เมื่อเทียบกับรายได้เฉลี่ยของผู้ที่จบปริญญาโทด้านวิทยาการคอมพิวเตอร์ใหม่ ๆ (จากมหาวิทยาลัยดี ๆ ที่นั่น) ก็อยู่ที่ประมาณ 5-7 หมื่นเหรียญเอง

แล้วงานประเภทไหนที่มีค่าจ้างสูง ๆ บ้าง ที่รายงานมาก็คือผู้ดูแลระบบฐานข้อมูล ผู้พัฒนาเว็บไซท์หรือเว็บมาสเตอร์ วิศวกรซอฟต์แวร์ และนักวิเคราะห์ระบบงาน ค่าแรงอยู่ที่ 90, 86, 85 และ 82 เหรียญต่อชั่วโมง ตามลำดับ



รูปที่ 1
การคอมไพล์
และรันโปรแกรมจาวา

เอาล่ะครับ ถึงตอนนี้ก็คงน่าสนใจขึ้นมาบ้างในตัวจาวา แต่ต้องเข้าใจว่าตัวเลขเหล่านี้เป็นเพียงภาพสะท้อนภาพหนึ่งซึ่งรายงานโดยทีมงานกลุ่มหนึ่งที่นั่น ซึ่งไม่จำเป็นว่าจะหมายถึงภาพรวมทั้งหมด (ใครสนใจขอให้ไปหารายละเอียดได้จากเว็บไซท์ที่บอกไว้ข้างต้นจะดีที่สุด) คงทราบกันดีว่าชาวอเมริกันเป็นพวกที่ชอบดูและทำสถิติกันมากที่สุด อย่างไรก็ตาม ตัวเลขเหล่านี้ก็คงช่วยให้เราเห็นภาพคร่าว ๆ ได้พอสมควรถึงความรุ่งเรืองของยุคไอทีที่อเมริกาและเกี่ยวพันมากับจาวาบ้าง

ถึงตอนนี้ ก็ขอกลับเข้าเรื่องที่เป็นทางเทคนิคเกี่ยวกับจาวาโดยตรงครับ

จาวาคืออะไร?

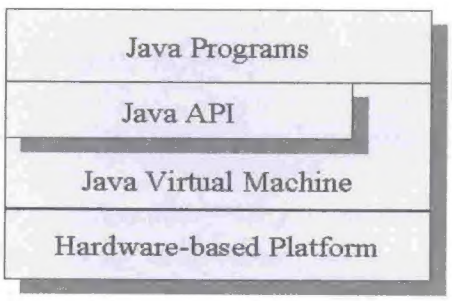
เพื่อความถูกต้องแน่นอน ต้องขอยกความหมายที่มาจากทีมผู้สร้างจาวาขึ้นมา นั่นคือทีมนักพัฒนาจากบริษัท Sun Microsystems (รัฐ California) เขาบอกว่า "Java is two things: a programming language and a platform." "Java is a high-level language that is all of the following: simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded and dynamic."

คงชัดเจนแล้วว่าจาวาคืออะไร จาวาเป็นทั้งภาษาคอมพิวเตอร์และเป็นแพล็ตฟอร์ม นั่นคือมีส่วนประกอบอื่น ๆ มากมายรวมกันในแง่ของการที่เป็นภาษา จาวาเป็นภาษาที่ถูกสร้างขึ้นโดยมีเงื่อนไขสำคัญหลายอย่างตามที่ทีมผู้สร้างเขานิยามไว้

Java is a Programming Language

โดยทั่วไปภาษาคอมพิวเตอร์อาจแบ่งได้เป็น 2 ลักษณะคือแบบคอมไพล์ (เป็นภาษาเครื่อง) และแบบ Interpret (คือตีความแล้วรันไปทีละคำสั่งด้วยตัวแปลภาษาหรือ Interpreter) แต่ภาษาจาวาเป็นแบบผสมทั้งสองชนิด กล่าวคือ โค้ดจาวาที่โปรแกรมเมอร์เขียนขึ้นจะต้องนำไปคอมไพล์ก่อน แล้วค่อยเอาผลลัพธ์ที่ได้ไปรันในลักษณะของการ Interpret อีกทีหนึ่ง ที่ว่า "คอมไพล์" นี้ไม่ได้คอมไพล์ให้เป็นภาษาเครื่องใดเครื่องหนึ่ง แต่จะให้ผลเป็น Java Bytecode ซึ่งเป็นโค้ดที่อิสระจากแพล็ตฟอร์มใด ๆ และจะถูกนำไป interpret โดยตัว Java interpreter ที่ว่าอิสระจากแพล็ตฟอร์มใด ๆ หมายถึงไบต์โค้ดนี้เป็นภาษากลาง ๆ ที่ใช้ได้กับคอมพิวเตอร์ทุกระบบที่มีตัว Java interpreter ตัว Java interpreter ที่ว่านี้เรียกว่า Java Virtual Machine (JVM) ซึ่งอาจเป็นซอฟต์แวร์ระบบหนึ่ง หรือส่วนประกอบหนึ่งในเว็บเบราเซอร์ หรือแม้แต่ฮาร์ดแวร์ก็ได้ ที่สามารถรับ bytecode ซึ่งมีรูปแบบมาตรฐานอยู่ได้ ไบต์โค้ดที่ว่านี้ทุกวันนี้ปกติจะอยู่ในรูปของไฟล์ ซึ่งมี extension ว่า .class เราจึงมักเรียกว่า class file หรือ Java class file เป็นต้น รูปที่ 1 แสดงขั้นตอนจากโค้ดภาษาจาวา (extension มาตรฐาน คือ .java) จะถูกคอมไพล์ให้เป็นไบต์โค้ด (หรือคลาสไฟล์) แล้วจึงไปรันด้วย Java interpreter

ด้วยการใช้ไบต์โค้ดเช่นนี้ ทำให้โค้ดจาวาโปรแกรมเดียวสามารถรันได้บนคอมพิวเตอร์แพล็ตฟอร์มใด ๆ ที่มี JVM ภายหลังจากคอมไพล์เป็น class file แล้ว ดังตัวอย่างเช่นปัจจุบัน มี JVM บนระบบปฏิบัติการ Windows 95/98/NT/2000, Sun Solaris Linux, HP/UX, Mac OS, SGI, IBM's AIX และ RS/6000 ฯลฯ



รูปที่ 2
สถาปัตยกรรมโดยรวมของจาวา

Java Program

โปรแกรมจาวามี 2 ลักษณะคือ Java Application และ Java Applet โดย Java Application เป็นโปรแกรมที่รันแบบ standalone อยู่บนจาวาแพล็ตฟอร์ม มีลักษณะคล้ายคลึงกับแอพพลิเคชันที่สร้างขึ้นจากภาษาอื่น ๆ ส่วน Java Applet เป็นโปรแกรมจาวาที่มีข้อกำหนดบางอย่างซึ่งทำให้มันรันได้บนเว็บเบราเซอร์ที่รัน Java Applet ได้ (Java-enabled browser)

นอกจากนี้ยังมีโปรแกรมจาวาพิเศษอีกอย่างหนึ่งคือ Servlet Servlet มีลักษณะคล้ายกับ Applet แต่ในขณะที่ Applet รันบนเว็บเบราเซอร์ Servlet จะรันที่ฝั่งของเซิร์ฟเวอร์ Servlet เป็นโปรแกรมที่เสริมการทำงานของตัวเว็บเซิร์ฟเวอร์ในลักษณะคล้าย ๆ กับการใช้สคริปต์ CGI ตัวอย่างเช่น ใช้ Servlet ประมวลผลข้อมูลที่รับมาทางฟอร์มแล้วบันทึกลงในฐานข้อมูล แต่ Servlet ใช้ทรัพยากรของเครื่องน้อยกว่าการใช้ CGI สามารถทำงานได้บนเครื่องและเว็บเซิร์ฟเวอร์หลายประเภท

Java Platform

แพล็ตฟอร์ม หมายถึง ระบบหรือสภาพแวดล้อมทางฮาร์ดแวร์และ/หรือซอฟต์แวร์ที่ใช้รันโปรแกรม แพล็ตฟอร์มทั่วไปมักเป็นส่วนประกอบร่วมของฮาร์ดแวร์และระบบปฏิบัติการ แต่จาวาแพล็ตฟอร์มไม่เป็นเช่นนั้น จาวาแพล็ตฟอร์มเป็นซอฟต์แวร์ล้วน ๆ ที่อยู่บนแพล็ตฟอร์มที่เป็นฮาร์ดแวร์อื่น ๆ แยกชั้นเพื่อความอิสระจากฮาร์ดแวร์นั่นเอง

ส่วนประกอบของ Java platform มี 2 ส่วนคือ Java Virtual Machine (หรือ JVM) (ซึ่งได้กล่าวถึงบ้างแล้ว) และ Java Application Programming Interface (หรือ Java API) ดังในรูปที่ 2

Java API เป็นชุดของซอฟต์แวร์คอมโปเนนต์ขนาดใหญ่ที่ช่วยเพิ่มความสามารถให้มากมาย เช่น กราฟิก การติดต่อบนเครือข่าย เป็นต้น Java API จะถูกจัดกลุ่มออกมาเป็น package หลาย package และมีการพัฒนาเพิ่มเติมและขยาย package เหล่านี้อย่างต่อเนื่อง

เมื่อ Java API ถูกแบ่งกลุ่มออกเป็น package และสามารถเพิ่มเติมหรือขยายได้มากมาย จึงได้มีการกำหนดชุด Java API ที่เป็นแกนหลักที่ Java Platform ทุกที่ (ไม่ว่าใครจะสร้าง) ควรจะต้องมีเรียกว่า core API ซึ่งช่วยการเขียนโปรแกรมในด้านต่าง ๆ ดังนี้

- พื้นฐานจำเป็น (The Essentials) ได้แก่ Objects, strings, threads, numbers, input and output, data structures, system properties, date and time ฯลฯ
- Applets ได้แก่ method และค่าคงตัวต่าง ๆ ที่ใช้ในการสร้าง Applets
- Networking ได้แก่ URLs, TCP หรือ UDP sockets, และ IP addresses เป็นต้น
- Internationalization ช่วยในการเขียนโปรแกรมเพื่อให้สอดคล้องกับผู้ใช้ในประเทศต่าง ๆ โปรแกรมสามารถปรับเข้าได้กับประเทศต่าง ๆ และใช้ภาษาที่เหมาะสมได้
- Security เรื่องความปลอดภัย ครอบคลุมทั้ง electronic signatures, การจัดการ public/private key, access control, และ certificates
- Software components ได้แก่ JavaBeans ซึ่งสามารถใช้งานร่วมกับสถาปัตยกรรมด้านคอมโปเนนต์ต่าง ๆ ได้ เช่น OLE/COM/Active-X ของ Microsoft และ Live Connect ของ Netscape
- Object serialization ช่วยเรื่อง lightweight persistence และการรับส่งข้อมูลโดยใช้ Remote Method Invocation (หรือ RMI)
- Java Database Connectivity (JDBC) ช่วยให้ใช้งานฐานข้อมูลชนิดรีเลชันแนลได้

ลักษณะสำคัญของจาวา

Simple หมายความว่าจาวามีโครงสร้างของภาษาที่ไม่ซับซ้อน ผู้ที่มีความรู้ด้านการเขียนโปรแกรมคอมพิวเตอร์มาบ้างจะสามารถเรียนรู้และใช้จาวาได้อย่างไม่ลำบาก โครงสร้างของภาษาส่วนใหญ่ได้ยืมมาจากภาษา C และ C++ แต่ได้ตัดส่วนที่ทำให้โปรแกรมซับซ้อนและเสี่ยงต่อการเกิดข้อผิดพลาด ลดโครงสร้างบางอย่างที่พอจะตัดและใช้อย่างอื่นแทนได้ เช่น union และ struct ใน C/C++ ก็ถูกยกเลิก และให้ใช้ class แทนทั้งหมด เป็นต้น

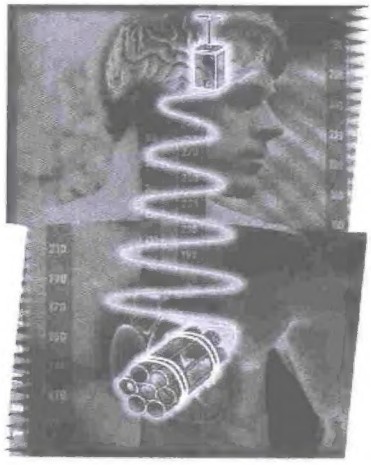
แต่จาวาได้มีการพัฒนามาอย่างต่อเนื่อง และมีการเพิ่ม API มากมายเพื่ออำนวยความสะดวกให้โปรแกรมเมอร์ จำนวน API ได้เพิ่มขึ้นจากประมาณ 200 ตัวในจาวาแพล็ตฟอร์มดั้งเดิมกลายเป็น 500 ในเวอร์ชัน 1.1 และกลายเป็น 1600 ใน Java 2 ในปัจจุบัน

Object-oriented จาวาถูกออกแบบมาให้เป็นเชิงวัตถุ (object-oriented) อย่างพอดี ๆ คือสามารถใช้ประโยชน์จากแนวคิด object-oriented ได้มาก แต่ก็ไม่ได้เป็นพวกสุดขั้ว (ที่ทุกอย่างแม้แต่เลขจำนวนเต็มก็ยังเป็นออบเจ็กต์ด้วย)

Java : จากเริ่มต้นจนถึงวันนี้ปี 2000

Distributed จาวาถูกออกแบบมาโดยคำนึงว่าต้องสามารถใช้เครือข่ายคอมพิวเตอร์โดยเฉพาะอย่างยิ่งอินเทอร์เน็ตได้อย่างดี จาวาถูกสร้างขึ้นมาตั้งแต่ประมาณปี 1991 คือพร้อม ๆ กับ World Wide Web แม้ว่าตอนแรกจะไม่ได้ตั้งใจจะสร้างเพื่อใช้กับคอมพิวเตอร์และอินเทอร์เน็ต แต่ต่อมาเมื่อมีการปรับมาใช้บนอินเทอร์เน็ตและ web จึงได้ยึดถือหลักแนวคิดข้อหนึ่งคือการใช้งานบน web ได้อย่างเต็มตัว จาวาแพล็ตฟอร์มมี package ชื่อ java.net ซึ่งช่วยการเขียนโปรแกรมที่ใช้เครือข่ายให้กลายเป็นเรื่องง่ายดายมาก นอกจากนี้ยังมี API เพื่อรองรับมาตรฐาน Common Object Request Broker Architecture (CORBA) ซึ่งเป็นมาตรฐานกลางสำหรับการสื่อสารระหว่างออบเจ็กต์บนเครือข่าย โดยมีเครื่องมือที่ช่วยต่อโปรแกรมจาวาเข้ากับ Interface Definition Language (IDL) ซึ่งเป็นภาษากลางที่ใช้ใน CORBA

Architectural-Neutral เท่าที่ผ่านมาโปรแกรมเกือบทั้งหมดจะถูกออกแบบและพัฒนาขึ้นโดยอิงอยู่กับระบบปฏิบัติการตัวใดตัวหนึ่ง ทำให้ไม่สามารถย้ายหรือนำโปรแกรมนั้น ๆ ไปใช้กับระบบปฏิบัติการอื่นได้ จึงได้เกิดแนวคิดความพยายามที่จะสร้างภาษาคอมพิวเตอร์ที่เขียนขึ้นทีเดียว ใช้ได้หลายที่และตลอดไป (Write once; Run anywhere, any time, forever) จาวาจึงมีหลักการที่จะให้โปรแกรมที่เขียนขึ้นด้วยภาษาจาวาสามารถใช้งานได้บนระบบปฏิบัติการใด ๆ ก็ตามที่มี Java Virtual Machine (หรือ JVM) ซึ่งจะรันอยู่เหนือระบบปฏิบัติการอีกทีหนึ่ง หากมีการเปลี่ยนแปลงที่สำคัญใด ๆ ในตัวระบบปฏิบัติการหรือฮาร์ดแวร์ ก็จะต้องปรับแก้เฉพาะเพียงตัว JVM เท่านั้น โดยหวังว่าจะไม่ต้องไปแก้โปรแกรมจาวาทั้งหมด ทุกวันนี้ JVM มีใช้สำหรับระบบปฏิบัติการที่ใช้กันแพร่หลายทุกระบบ เช่น Sun Solaris, HP/UX, Linux, Mac OS, Microsoft Windows NT/2000/98/95, SGI, IBM's AIX และ RS/6000 เป็นต้น อย่างไรก็ตาม การที่ออกแบบให้ระบบไม่ขึ้นอยู่กับเครื่องใด ๆ ก็มีข้อด้อยตรงที่ไม่สามารถดึงความสามารถพิเศษของแต่ละเครื่องออกมาใช้ได้เต็มที่



Interpreted โปรแกรมจาวาจะถูกคอมไพล์ระดับหนึ่งให้เป็นไบต์โค้ดซึ่งสามารถนำไปรันได้บนเครื่องใดก็ได้ที่มี JVM ด้วยสถาปัตยกรรมเช่นนี้ ขั้นตอนการลิงก์โปรแกรมจึงง่ายและสามารถขยายเพิ่มเติมโปรแกรมได้สะดวก ช่วยส่งเสริมแนวทางการพัฒนาระบบงานแบบโปรโตไทพ์ และแนวทาง Rapid Development จึงเป็นเรื่องธรรมชาติ แตกต่างจากแบบเดิม ๆ ที่แต่ละโปรแกรมต้องคอมไพล์แล้วลิงก์รวมกันแล้วจึงทดสอบ

Robust and Secure ภาษาจาวามีโครงสร้างหลายอย่างที่ลดความเสี่ยงจากการใช้หน่วยความจำผิด ทั้งจากการจองใช้หน่วยความจำแล้วไม่คืน หรือการอ้างถึงหน่วยความจำนอกขอบเขต เป็นต้น ซึ่งเป็นปัญหามากกับโปรแกรมภาษา C/C++ จาวามีการจัดการข้อผิดพลาดต่าง ๆ ที่เกิดขึ้น เช่น การหารด้วยเลขศูนย์ หาไฟล์ไม่พบ ส่งข้อมูลผ่านเครือข่ายไม่ได้ และอื่น ๆ อีกมากมาย ข้อผิดพลาดเหล่านี้โปรแกรมเมอร์สามารถตรวจจับและจัดการได้ด้วยกลไกการจัดการข้อผิดพลาด (Exception Handling) ที่จาวามีให้ เนื่องจากจาวาถูกออกแบบสร้างขึ้นมาให้ใช้กับสภาพแวดล้อมแบบ Distributed เรื่องความปลอดภัยจึงกลายเป็นเรื่องที่สำคัญอย่างยิ่ง การออกแบบจาวาจึงได้ให้ความสำคัญมากกับเรื่องนี้ทั้งในส่วนโครงสร้างของภาษาจาวาและในส่วนของ JVM

เช่นโปรแกรมที่รันอยู่ใน JVM จะใช้งานทรัพยากรใด ๆ ของเครื่องนั้นได้โดยต้องผ่าน API ของ JVM เท่านั้น จะไปใช้งานตรง ๆ ไม่ได้ และจาวานี้ type safe นั่นคือไม่สามารถเล่นกลกับชนิดของข้อมูลได้ กล่าวคือ ตัวอย่างเช่น คุณไม่สามารถทาบโครงสร้างข้อมูล (เช่น class) ของระบบด้วยโครงสร้างข้อมูลของคุณเองเพื่อที่จะได้ (แอบ) แก้ไขข้อมูลระบบนั้นโดยการเรียกใช้โครงสร้างข้อมูลของคุณที่เอาไปทาบ

JVM จะรันโค้ดใด ๆ ที่ไม่มีการรับรอง อยู่ภายใน sandbox ซึ่งมีการจำกัดสิทธิต่าง ๆ มากมายเพื่อความปลอดภัยของเครื่องและตัว JVM เช่น ไม่อนุญาตให้เขียนไฟล์ในเครื่องนั้น หรืออ่านค่า properties บางอย่างของระบบ และก่อนจะ interpret ไบต์โค้ด ก็จะมีการตรวจสอบไบต์โค้ดนั้นก่อนว่าปลอดภัยตามมาตรฐานหนึ่ง เช่น ตรวจว่าแต่ละคำสั่งมีการใช้โค้ดคำสั่ง (operation code) ที่ถูกต้อง

Threaded นอกจากนี้จาวายังรองรับการสร้างโปรแกรมแบบหลาย Thread ซึ่งให้โปรแกรมเมอร์เขียนโปรแกรมขึ้นมาโปรแกรมหนึ่งที่เสมือนกับสามารถแยกการทำงานออกเป็นส่วนย่อย ๆ ขนานกันไป (เรียกแต่ละส่วนว่า Thread) เช่น สามารถทำ animation บนหน้าจออยู่ในขณะที่กำลังดาวน์โหลดไฟล์จากเครือข่าย จาวาแพล็ตฟอร์มช่วยให้โปรแกรมเมอร์สร้างโปรแกรมลักษณะเช่นนี้ได้ในระดับการเขียนโค้ด โดยการเรียกใช้คลาส Thread ซึ่งเป็นส่วนหนึ่งของไลบรารีมาตรฐานของจาวา และนอกจากนี้คลาสนี้ยังช่วยในเรื่องการประสานงานกันระหว่าง Thread ย่อย ๆ เหล่านี้ด้วย

Dynamic ในขั้นตอนการลิงก์คลาสเข้ากันเป็นไปอย่าง Dynamic กล่าวคือ ข้อมูลเกี่ยวกับชนิดและการเรียกใช้คลาสจะถูกตรวจสอบและจัดการในระหว่างที่โปรแกรมกำลังรันอยู่ ไม่ใช่ระหว่างคอมไพล์หรือก่อนรัน (เช่น ในระบบที่เรียกว่า Static Linking) ส่วนของโปรแกรมจึงสามารถนำมาเรียกใช้ได้ตามต้องการจากหลาย ๆ แหล่ง รวมทั้งจากเครื่องอื่นบนเครือข่าย ช่วยให้การแก้ไขโปรแกรมทำได้อย่างสะดวกและทันที

High Performance จากการที่จาวามีขั้นตอนการรันแบบ interpretation ซึ่งปกติแล้วจะช้ากว่าการรันโปรแกรมที่คอมไพล์เป็นภาษาเครื่องแล้วนับสิบ ๆ เท่า แต่จาวาไบต์โค้ดที่สร้างขึ้นมาจะมีความใกล้เคียงกับภาษาเครื่อง การ interpret จะทำได้รวดเร็วมาก หากเทียบกับภาษาคอมพิวเตอร์ประเภท interpret ด้วยกัน ตัวจัดการ (เก็บกวาด) หน่วยความจำที่ใช้แล้วหรือ Automatic Garbage Collector จะแอบทำงานอย่างอัตโนมัติและเงียบ ๆ โดยพยายามไม่ไปรบกวนการทำงานของโปรแกรมจาวาที่โปรแกรมเมอร์เขียนขึ้น และแอพพลิเคชันที่ต้องการใช้ CPU คำนวณมาก ๆ อาจออกแบบโค้ดส่วนที่ใช้ CPU มาก ๆ เขียนขึ้นด้วยภาษาเครื่อง แล้วมาต่อกับโค้ดจาวาที่เหลือได้ นอกจากยังมีคอมไพเลอร์ Just-In-Time (JIT) ซึ่งเสริมตัว JVM เพื่อแปลงไบต์โค้ดให้กลายเป็นออบเจ็กต์โค้ดของเครื่องนั้น และรันได้รวดเร็วกว่าการ interpret ทีละคำสั่ง

และผู้ผลิตคอมพิวเตอร์หลายรายก็ยังได้พัฒนาคอมไพเลอร์ภาษาจาวาให้กลายเป็นภาษาเครื่องสำหรับเครื่องนั้น ๆ และสามารถรันได้โดยไม่ต้องใช้ JVM มาตรฐาน ตัวอย่างเช่น คอมไพเลอร์ IBM High Performance Java บนระบบ Windows, OS/2, AIX, AS/400 และ System 390 ซึ่งทำให้โปรแกรมสามารถรันได้เร็วกว่าแต่ก็ต้องรันบนเครื่องนั้น ๆ เฉพาะเท่านั้น

จาวาช่วยอะไรให้โปรแกรมเมอร์?

ด้วยคุณลักษณะสำคัญที่กล่าวไว้ข้างต้น อาจมองเห็นได้ว่าจาวาช่วยโปรแกรมเมอร์ในด้านต่าง ๆ ได้แก่ ช่วยให้เริ่มต้นได้เร็ว แม้ว่าภาษาจาวาจะเป็น object-oriented ที่มีประสิทธิภาพ แต่ก็ง่ายในการเรียนรู้ โดยเฉพาะกับผู้ที่รู้จัก C หรือ C++ มาแล้ว จาวาช่วยให้เขียนโค้ดน้อยลง จากการเปรียบเทียบจำนวน class จำนวน method และอื่น ๆ พบว่า โปรแกรมที่เขียนด้วยจาวามีขนาดเพียงประมาณ 1 ใน 4 ของโปรแกรมเดียวกันในภาษา C++ จึงช่วยให้การพัฒนาโปรแกรมรวดเร็วขึ้น หากเขียนภาษาจาวาโดยไม่เรียกใช้ class หรือ method ใดที่อยู่นอกมาตรฐาน ก็รับรองได้ว่า ไบต์โค้ดนั้นจะอิสระจากแพล็ตฟอร์ม นั่นคือรันได้ทันทีและได้ผลเหมือนกันบนแพล็ตฟอร์มมากมาย

นอกจากนี้การเขียนโปรแกรมบนอินเทอร์เน็ตทำได้อย่างสะดวก เนื่องจากมีอยู่ใน core API แล้ว และยังสามารถกระจายโปรแกรมได้อย่างรวดเร็ว โดยแนวทางที่ Applet จะถูกดึงจากเซิร์ฟเวอร์ทำให้สามารถแก้ไข class และใช้ได้ทันที โดยไม่ต้องคอมไพล์ทั้งโปรแกรมใหม่

แหล่งข้อมูลสำคัญสำหรับจาวา

ถึงตอนนี้หลายท่านคงเริ่มสนใจหรือเริ่มงงมากขึ้นว่าทั้งหมดที่เล่ามาเป็นจริงหรือไม่ ก็คงถึงเวลาที่จะบอกเล่าถึงแหล่งข้อมูลฟรีบนอินเทอร์เน็ตที่น่าบันทึกไว้สำหรับผู้ที่ต้องการศึกษาจาวาอย่างถ่องแท้ ก่อนที่จะไปสู่ครึ่งหลังของตอนนี้ซึ่งกล่าวถึงเทคโนโลยีมากมายเกี่ยวกับจาวา รายชื่อต่อไปนี้เป็นรายชื่อเว็บไซท์เท่าที่นึกออกและใช้บ่อย ๆ เวลานี้เท่านั้น ซึ่งความจริงยังมีที่อื่นอีกมากมาย

- แหล่งกำเนิด (ต้องไม่พลาด) java.sun.com และ www.sun.com
- ต้องการดาวน์โหลด Java Development Kit ของ Sun มาใช้เขียนโปรแกรมจาวา ก็ต้องไปที่ http://java.sun.com/j2se/ หรือ http://java.sun.com/j2se/1.3 โดยตรง JDK มีแต่คอมไพเลอร์และจาวาแพล็ตฟอร์มให้ แต่ไม่มี Integrated Development Environment (IDE) ให้ใช้สะดวก ๆ หากต้องการ IDE สำหรับจาวาให้ลองดู JBuilder 3.5 ของ Inprise, VisualAge for Java 3 ของ IBM, AnyJ ของ Netcomputing หรือ Visual Cafe ของ Symantec เป็นต้น แต่ IDE เหล่านี้ส่วนใหญ่ไม่ใช่ของฟรี แต่เราสามารถดาวน์โหลด Trial version มาทดลองใช้ได้
- อยากได้ Tutorial เก็บไว้ ก็ต้องไปที่ http://java.sun.com/docs/books/tutorial/ ซึ่งสามารถดูจากอินเทอร์เน็ตก็ได้ หรือจะดาวน์โหลดมาเก็บดูเองก็ได้เช่นกัน รับรองว่าคุ้มจริง ๆ
- เว็บไซท์ที่ไม่ควรพลาดอย่างยิ่ง : gamelan.earthweb.com (เดิม www.gamelan.com) และ www.devx.com (ซึ่งมีเรื่องราวความรู้เกี่ยวกับคอมพิวเตอร์หลากหลายเรื่องราวและมากมายจริง เช่น C++, ASP, VB, Linux, Java, XML และลิงก์ไปยังนิตยสารอีกมากมาย)
- นิตยสารฟรีบนเว็บ เช่น : www.javaworld.com, www.javadevelopersjournal.com, www.java-pro.com, www.servletcentral.com หรือ www.javareport.com เป็นต้นดี ๆ ทั้งนั้น
- รวมทั้งเรื่องราวเกี่ยวกับจาวาที่บริษัทยักษ์ใหญ่มี เช่น www.ibm.com/developer/java, www.microsoft.com/java, www.oracle.com, www.borland.com/jbuilder เป็นต้น

ประวัติที่มาและวิวัฒนาการของจาวา

จาวานี้ความจริงเริ่มต้นจากการที่บริษัท Sun Microsystems ออกแบบซอฟต์แวร์สำหรับใช้งานบนเครื่องใช้อิเล็กทรอนิกส์ที่ใช้ชิปราคาถูก ๆ โดยทดลองสร้างคอมพิวเตอร์ขนาดมือถือชื่อ *7 ซึ่งจะใช้ควบคุมเครื่องใช้ไฟฟ้าภายในบ้าน โดยมี James Gosling เป็นหัวหน้าทีม โครงงานนี้ไม่ได้นำออกจำหน่ายแต่อย่างไร แต่ทีมพัฒนานี้ได้เห็นว่าซอฟต์แวร์ควบคุมดังกล่าว (ซึ่งชื่อว่า Oak) สามารถประยุกต์มาใช้กับงานอื่นได้ นั่นคือบนอินเทอร์เน็ตและเว็บ ซึ่งต้องการคุณสมบัติสำคัญสองอย่างคือ ความกะทัดรัดและความง่าย จึงได้มีการปรับปรุงพัฒนา Oak และเปลี่ยนชื่อมาเป็นจาวา (คงเนื่องจากว่าเวลานั้นมีภาษาคอมพิวเตอร์ที่ชื่อ Oak อยู่ก่อนแล้ว)

Sun Microsystems ได้ประกาศตัวจาวาเป็นทางการในกลางปี 1995 โดยมีชุดซอฟต์แวร์เพื่อพัฒนาโปรแกรมจาวาหรือที่เรียกคุ้นเคยกันดีว่า Java Development Kit หรือ JDK เวอร์ชัน 1.0 ผลคือได้รับการต้อนรับอย่างดีจากวงการคอมพิวเตอร์ เว็บเบราเซอร์ที่นิยมส่วนมากได้พัฒนา JVM ไว้ในตัว

ต่อมาในปี 1997 Sun Microsystems ได้ประกาศ JDK1.1 ซึ่งมีการพัฒนาเปลี่ยนแปลงมากมาย เช่น การจัดการ event และเริ่มต้นมี JavaBeans ซึ่งเป็นเป็นซอฟต์แวร์คอมโปเนนต์ที่เขียนด้วยจาวาขึ้นเป็นครั้งแรก เว็บเบราเซอร์ส่วนใหญ่ได้ปรับปรุง JVM ในตัวให้เป็นเวอร์ชัน 1.1 ด้วย

ต่อมา ในช่วงปี 1997 ถึง 1999 Sun Microsystems ได้พัฒนาจาวาอย่างมาก เช่น ออก JDK1.1 ย่อย ๆ หลายรุ่น ออก Java Foundation Classes (JFC) ในปี 1998 เพื่อให้แยกดาวน์โหลดได้ และปี 1998 นี้ก็ออก API สำหรับ Servlet ในชุด Java Servlet Development Kit (JSDK) และ Java Server Pages (JSP) เว็บเซิร์ฟเวอร์ของหลายบริษัทก็ได้พัฒนาให้รองรับ JSP ด้วย

ช่วงปลายปี 1998 Sun Microsystems ได้ประกาศ Java 2 ออกมาเป็นชื่อเรียกชุดซอฟต์แวร์รุ่น JDK1.2 และชุดซอฟต์แวร์ JDK1.2 นี้ได้ถูกเรียกใหม่ว่าเป็น Java 2 Software Development Kit, Standard Edition

JFC ได้ถูกรวมเป็นส่วนหนึ่งของ Java 2 ซึ่งรวมเอาเทคโนโลยีมากมายเข้ามา ได้แก่ JDBC เพื่อใช้งานฐานข้อมูล หรือการใช้งาน CORBA เป็นต้น

กลุ่มบริษัทหลายบริษัทได้ร่วมมือกันออกมาตรฐาน Enterprise Java Beans (EJB) ออกมาครั้งแรก ซึ่งเป็นมาตรฐานสำหรับสถาปัตยกรรมของซอฟต์แวร์คอมโปเนนต์ และบริษัทจำนวนมากได้ปรับปรุงและจำหน่าย Enterprise Java Servers ที่รองรับมาตรฐาน EJB

เดือนมิถุนายน 1999 Sun Microsystems ได้ประกาศอย่างเป็นทางการถึง Java 2 Platform โดยมีถึง 3 ขนาดจากเล็กไปใหญ่ (ขนาดที่เล็กกว่าจะเป็นส่วนหนึ่งของขนาดที่ใหญ่ลำดับต่อไป) คือ Java 2 Platform, Micro Edition (J2ME) ใช้สำหรับอุปกรณ์ขนาดเล็กที่ใช้จาวาได้ เช่น Palm Pilot, เพจเจอร์ นาฬิกา และอื่น ๆ Java 2 Platform, Standard Edition (J2SE) ใช้กับ Applet และ application ที่นิยมใช้กันแพร่หลายด้วย JDK มาตรฐาน และ Java 2 Platform, Enterprise Edition (J2EE) ซึ่งรวม Enterprise API เข้าไปให้กลับกลายเป็นแพล็ตฟอร์มสมบูรณ์แบบสำหรับการใช้งานคอมโปเนนต์ฝั่งเซิร์ฟเวอร์ระดับองค์กรด้วยภาษาจาวา

เดือนสิงหาคม 1999 Sun Microsystems ได้ออกข้อกำหนด JSP1.0 และชุดซอฟต์แวร์ JavaServer Web Development Kit ออกมาใช้งาน

และปัจจุบัน กลางปี ค.ศ. 2000 Sun Microsystems ได้ออก Java Development Kits หรือ JDK1.3 ออกมาแล้ว

การเปลี่ยนแปลงใน JDK

การเปลี่ยนแปลงปรับปรุงใน JDK ที่ผ่านมา อาจพอสรุปได้ดังนี้

สิ่งที่เพิ่มขึ้นใน 1.1

- JavaBeans ซึ่งเป็นซอฟต์แวร์คอมโปเนนต์ที่เขียนด้วยจาวาและสามารถสร้างและใช้ทูลที่เป็นแบบรูปภาพทั้งหลาย (visual tools) JavaBeans อาจจะเป็นคอมโปเนนต์ง่าย ๆ อย่างปุ่มกด หรือไดอะล็อกบ็อกซ์ หรืออาจจะซับซ้อนอย่างสเปรดชีตหรือปฏิทินเลยก็ได้
- Serialization ซึ่งช่วยให้เราสามารถเก็บสถานะของออบเจ็กต์และนำมาใช้ใหม่ได้ภายหลัง
- Remote Method Invocation (RMI) ซึ่งช่วยให้ออบเจ็กต์ของจาวาสามารถเรียกใช้ออบเจ็กต์จาวาอื่นที่อยู่ในคอมพิวเตอร์อีกเครื่องหนึ่งบนเครือข่ายได้ (โดยไม่ต้องใช้คำสั่งประเภท socket หรือ stream) ซึ่งเป็นส่วนประกอบสำคัญอันหนึ่งในการสร้าง distributed application
- Java Database Connectivity (JDBC) ซึ่งช่วยให้โปรแกรมจาวาสามารถใช้งานฐานข้อมูลได้มากมายในลักษณะของการส่งคำสั่ง SQL ไปและสามารถรับผลลัพธ์กลับมาใช้งานต่อได้
- Java Native Language (JNI) ซึ่งเป็นวิธีหนึ่งที่จะให้โปรแกรมติดต่อกับไลบรารีที่เขียนขึ้นจากภาษาอื่น ๆ ได้
- Reflection ซึ่งช่วยในการพิจารณาตรวจดู method, constructors และ fields ของออบเจ็กต์จาวาได้ในระหว่างรันโปรแกรม

• ระบบรักษาความปลอดภัย มีให้หลายอย่าง เช่น digital signature, message digest, access control list และการสร้าง key

• ปรับปรุงการจัดการ event ที่เกิดขึ้นจากคอมโปเนนต์ทาง GUI ให้ดีขึ้น

• Inner class ซึ่งช่วยให้สามารถประกาศ class หนึ่งภายในอีก class หนึ่งได้

สิ่งที่เปลี่ยนแปลงจาก JDK1.1 ไป JDK1.2

1. ปรับปรุงระบบรักษาความปลอดภัย กลไกการควบคุมการใช้งานวัตถุใด ๆ หรือ Access Control ใน JDK1.2 เป็นแบบใช้ policy (หรือนโยบาย) ปรับแต่งได้ง่าย และควบคุมได้ละเอียดยิ่งขึ้น โดยใช้หลักการของ "policy" และ "permission" (นโยบายและรายการอนุญาต) เมื่อโค้ดถูกโหลดขึ้นมา มันจะได้รับ permission ซึ่งขึ้นอยู่กับ policy ด้านความปลอดภัยที่บังคับใช้ในเวลานั้น permission แต่ละค่าจะกำหนดว่าจะอนุญาตให้ใช้ทรัพยากรได้หรือไม่อย่างไร เช่น กำหนดให้อ่านหรือเขียนไฟล์หรือไดเรคทอรีอันหนึ่งได้หรือไม่ หรืออนุญาตให้ติดต่อกับโฮสต์และพอร์ตชุดหนึ่ง ๆ ได้หรือไม่

policy หรือนโยบายนี้ อาจกำหนดได้จากไฟล์ที่เก็บนโยบาย ซึ่งแก้ไขได้ โค้ดต่าง ๆ จะไม่สามารถใช้งานทรัพยากรต่าง ๆ ที่ควบคุมได้ หากไม่ได้กำหนดว่าอนุญาตให้ใช้ได้ การควบคุมโดยการใช้ policy และ permission นี้ใช้ได้กับทั้ง Applet, application และก็ bean ทั้งหลายด้วย

Certificate interface ช่วยให้โปรแกรมเมอร์สามารถอ่าน certificate กำหนดนโยบายเกี่ยวกับ certificate ในไซท์ตนเอง และจัดการฐานข้อมูลเก็บ certificate ได้ และยังมีการรับรอง x.509 v3 ซึ่งเป็นข้อตกลงเกี่ยวกับ certificate ที่นิยมใช้กันมากด้วย

เครื่องมือเกี่ยวกับการรักษาความปลอดภัย ได้แก่ keytool, jar, jarsigner และ policytool JDK1.2 นี้มีซอฟต์แวร์ทูล keytool ให้ใช้เพื่อช่วยในการสร้างคีย์ลับ และ x.509 v1 certificate ซึ่งจะช่วยในเรื่องลายเซ็นดิจิตอลให้แก่แอพพลิเคชันและ Applet โปรแกรม jarsigner มีไว้กำกับลายเซ็นเพื่อยืนยัน JAR ไฟล์ (Java Archive Format)

2. Swing (JFC) Swing เป็นคอมโปเนนต์ใหม่ด้าน GUI ที่ "Pluggable Look and Feel" หมายถึงช่วยให้โปรแกรมเมอร์กำหนดรูปแบบการแสดงแบบที่ขึ้นกับระบบปฏิบัติการ (เช่น MacOS, Motif หรือ MS Windows) หรือเป็นแบบมาตรฐานเดียวกัน (สไตล์ Java) ได้โดยสามารถสร้างหรือแก้ไขรูปแบบการแสดงเองได้ Swing มีพื้นฐานอยู่บน JDK1.1 และออกแบบขึ้นมาเป็น "100% Pure Java" (ซึ่งเป็นมาตรฐานที่รับรองโปรแกรมที่ผู้ใช้จะพบ user interface ลักษณะเดียวกันหมดไม่ว่าจะใช้งานบนแพล็ตฟอร์มไหน) และนอกจากการใช้งาน Swing สามารถใช้คีย์บอร์ดโดยไม่ต้องใช้เมาส์เลยก็ได้ด้วย

3. Java 2D (JFC) เป็นกลุ่มของคลาสที่ขยาย package java.awt และ java.awt.image สำหรับงานกราฟิก 2 มิติ และรูปภาพที่ก้าวหน้าขึ้น เช่น line art การกำหนดและเปลี่ยนสี และคำสั่งมากมายในการแสดงรูปภาพ

4. Accessibility (JFC) Accessibility API ช่วยให้ JFC และ AWT คอมโปเนนต์ทั้งหลายเชื่อมต่อกับเทคโนโลยีที่ช่วยให้ใช้คอมพิวเตอร์ได้ง่ายขึ้น (นึกถึง Accessibility ใน control panel ของ Windows ก็ได้) เช่น ตัวขยายจอภาพ (screen magnifier) speed recognition และการใช้คีย์บอร์ดแทนเมาส์

5. Drag and Drop ช่วยด้านการถ่ายข้อมูลด้วยการ drag and drop ทั้ง application ของจาวาด้วยกันเอง หรือระหว่างจาวากับภาษาอื่นก็ได้ รวมทั้งภายใน application เดียวกัน

6. Application Service ได้แก่ การกำหนด mouse cursor เองได้ DebugGraphics ช่วยในการทดสอบว่าคอมโปเนนต์ Swing ที่สร้างขึ้นมาวาดถูกต้องหรือไม่ Undo ช่วยในการย้อนคำสั่งที่ทำไปแล้ว Bounded Range Model ช่วยกำหนดค่าที่มีของเซต (เช่น ใช้ slider, scrollbar หรือ progress bar) และ Repaint Batching ช่วยการควบคุมการวาดพื้นที่บนหน้าจอใหม่ให้ควบคุมได้ดีขึ้น

7. Collections Framework เป็นกลุ่ม API ที่ช่วยทำงานเกี่ยวกับกลุ่มข้อมูลต่าง ๆ เช่น Dictionary, Vector, Stack, LinkedList เป็นต้น โดยช่วยให้โปรแกรมเมอร์ใช้งานข้อมูลเหล่านี้ได้อย่างสะดวกและมีประสิทธิภาพมาก โดยใน framework นี้มีสิ่งต่าง ๆ ทั้งในรูปของ interfaces และอัลกอริธึม

8. Extension Framework เป็นส่วนประกอบที่ช่วยในการดาวน์โหลดและติดตั้งชุด package ส่วนที่เพิ่มเติมจากจาวาแพล็ตฟอร์มมาตรฐาน โดยส่วนขยายเพิ่มเติมนี้อาจมาจาก Sun เองหรือบริษัทอื่นก็ได้

9. เพิ่มความสามารถให้ RMI เช่น เพิ่ม persistent reference ให้กับ remote object ซึ่งใช้ช่วยในการเรียกใช้ออบเจ็กต์ และ Custom Socket Factory ช่วยการทำ RMI บนโปรโตคอลอย่างเช่น SSL เป็นต้น

10. Java 2 มี ORB ให้ใช้กับมาตรฐาน Common Object Request Broker Architecture (CORBA) ซึ่งเป็นมาตรฐานสากลสำหรับการเชื่อมต่อและการทำงานพร้อมกันระหว่างออบเจ็กต์ทั้งหลายบนเครือข่ายและคอมไพเลอร์ idltojava ซึ่งจะช่วยสร้างโค้ดขึ้นมาจากสเปก IDL โดยจะได้ส่วน stub ของฝั่ง client และส่วน skeleton ของฝั่งเซิร์ฟเวอร์

11. เพิ่มเติมเทคโนโลยีด้าน JDBC ปรับปรุงใหม่เป็น JDBC 2.0 ให้เป็นส่วนเสริมในจาวาแพล็ตฟอร์ม

12. ขยายความสามารถด้านเสียง มี sound engine ช่วยให้ใช้ไฟล์เสียงแบบต่าง ๆ เช่น WAV, AIFF, AU, MIDI และ RMF ได้คุณภาพเสียงดีขึ้นมาก และใน java.applet.Applet มี method ที่ช่วยในการสร้าง AudioClips ด้วย

13. เพิ่มเติมด้านประสิทธิภาพ เช่น มี Just In Time (JIT) compiler ในชุด JDK ทั้งสำหรับ Windows และ Solaris ซึ่งจะช่วยเพิ่มประสิทธิภาพในเวลารันโปรแกรมจาวาได้มาก และใน Solaris ยังสามารถใช้ประสิทธิภาพของ multithread ซึ่งมีในตัวระบบปฏิบัติการ Solaris อยู่แล้วมาเพิ่มประสิทธิภาพของ thread ในจาวาและเลี่ยงการ map I/O system call ซึ่งประสิทธิภาพไม่ดี นอกจากนี้การจัดสรรหน่วยความจำและการจัดการหน่วยความจำที่ใช้แล้ว (Garbage Collection) ยังใช้ thread-local heap cache ที่ช่วยลดการ lock ในการขอ heap ส่วนใหญ่ ทำให้การขอหน่วยความจำรวดเร็วขึ้นมาก และการคืนหน่วยความจำก็ใช้เวลาน้อยลง

14. ประสิทธิภาพของ Remote Method Invocation (RMI) ดีขึ้น การจัดการ Garbage Collection ในส่วนที่อยู่คนละเครื่องเกิดการติดต่อบนเครือข่ายน้อยลง และลดการเกิด bottleneck ลงด้วย รวมทั้งการปรับปรุง dynamic class loading ของ RMI เป็นต้น

15. นอกจากนี้ยังมีการปรับปรุงในรายละเอียดอีกหลายส่วน เช่น การคำนวณเลข floating point ให้มีประสิทธิภาพดีขึ้นใน CPU ที่แพร่หลาย text คอมโปเนนต์สามารถรับอักษรภาษาจีน ญี่ปุ่น และเกาหลีได้ (โดยใช้ชุดของคีย์ต่อตัวอักษร 1 ตัว) การปรับปรุงทูล javac, java และ javadoc มี interface สำหรับ Debugger และ Profiler ให้ การปรับปรุง JavaBeans การควบคุมเวอร์ชันให้ application และ Applet, Serializable Fields API ช่วยกำหนดและใช้งาน serializable fields เป็นต้น

สิ่งที่ตัดไปโดย Java1.1 และ Java2

ใน Java1.1 ได้ยุติการใช้งาน method หลายอย่าง เช่น บางส่วนของคลาส Date ของเก่า แต่การตัดนี้ไม่ใช่ตัดทิ้งไปเลย แต่ได้เปลี่ยนแปลงเป็นของใหม่ที่ดีขึ้น (ใช้คลาส Date Format และ Calendar แทน) ส่วนใน Java2 ได้มีการตัดบางอย่างใน Java1.1 ออกบ้าง แม้จะไม่มากนัก ตัวอย่างเช่น method suspend(), resume() และ stop() ของคลาส Thread ก็ไม่ควรจะใช้อีกต่อไป เป็นต้น

ทิ้งท้าย : ลงมือเขียนโค้ด?

ถึงตอนนี้หลายท่านที่เพิ่งรู้จักจาวาจากบทความนี้ คงสงสัยอยากจะลองลงมือเขียนโปรแกรมจาวาขึ้นมาสักโปรแกรมหนึ่ง ความจริงแล้วการเขียนโปรแกรมเบื้องต้นประเภท Hello World ด้วยจาวาสามารถหาอ่านได้จากหนังสือที่เกี่ยวกับจาวาเบื้องต้นทุกเล่มอยู่แล้ว จึงขอละไว้ ไม่กล่าวถึงในที่นี้ ลองไปหาดูตามร้านหนังสือต่าง ๆ หรือหากมีอินเทอร์เน็ตต่ออยู่ ก็ไปเข้าเว็บไซท์ที่กล่าวไว้ข้างต้นได้เลย เช่น ไปดาวน์โหลด Java Development Kit 1.3 ของ Sun มาใช้จาก http://java.sun.com/j2se/ หรือ http://java.sun.com/j2se/1.3

ซึ่ง JDK นี้มีจาวาคอมไพเลอร์ชื่อ javac ไว้ให้คอมไพล์โค้ดจาวาที่เขียนขึ้น และเมื่อจะรันทดสอบก็ต้องใช้ตัว JVM (หรือ Java Runtime Environment) โดยใช้โปรแกรมที่ชื่อ java หากต้องการทดสอบ Applet ที่เขียนขึ้นก็ใช้โปรแกรมชื่อ appletviewer ได้ ซึ่งทั้งหมดนี้ (และโปรแกรมอื่น ๆ อีกมาก) ส่วนใหญ่เป็นคำสั่งแบบ MSDOS command line และอยู่ในไดเรคทอรีชื่อ <jdk__home>/bin เมื่อ <jdk__home> เป็นไดเรคทอรีที่คุณติดตั้ง JDK ไว้ ซึ่งส่วนตัวขอแนะนำว่าน่าจะลงไว้ที่ root เลย เช่น C:\jdk1.3 หรือ D:\jdk1.3 ก็ได้ เนื่องจากคู่มือส่วนใหญ่จะใช้ชื่อลักษณะนี้

หากมีเวลา คราวหน้าจะได้กล่าวถึงมาตรฐานหรือธรรมเนียมการเขียนโปรแกรมจาวาและ/หรือความแตกต่างหรือการเปลี่ยนการเขียนโค้ด C++ เป็นจาวาสำหรับโปรแกรมเมอร์ที่ใช้ C++ อยู่ ท้ายนี้ขอให้ผู้ที่ตั้งใจจะลองใช้จาวาจริงประสบผลสำเร็จเร็ว ๆ ขอบคุณครับ

อ้างอิง

- D. Flanagan "Java in a Nutshell", O'Reilly & Associates, 1996.
- H. M. Deitel & P. J. Deitel, "Java How to Program", 2ed., PHI, 1998.
- J. Wigglesworth, P. Lumby, "Java Programming Advanced Topics", Course Technology, 2000.
- P. Naughton, H. Schildt, "Java 2 The Complete Reference", MGH, 1999.
- Sun Microsystems, "The Java Language: A White Paper", 1996.
- Sun Microsystems, "Javatm2 SDK, Standard Edition Documentation v1.3" at http://java.sun.com/j2se/1.3/docs.
- Sun Microsystems, "JDK v1.2 Summary of New Features" at http://java.sun.com/products/jdk1.2/docs/relnotes/features.html.